Text Processing in Cocoa

Session 112

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

At the convention center, a movie will be placed here

Text Processing in Cocoa

Douglas Davidson
Software Engineer

Introduction

- Text in Cocoa is always Unicode-based
- NSString is the fundamental object
- Deal with strings and substrings, not characters or bytes
- Work with the framework

What You'll Learn

- Reading and writing text
- Processing text
- Checking text

Reading and Writing Text

NSString Methods

- The text is in a plain-text file whose encoding is known
- You wish to read or write the entire file at once

```
NSError *error;
NSString *string = [[NSString alloc] initWithContentsOfURL:url
                                                  encoding:NSUTF8StringEncoding
                                                  error:&error];

[string writeToURL:url atomically:YES
    encoding:NSUTF8StringEncoding error:&error];
```

Encodings

- Unicode encodings
  - UTF-8 (for compatibility with ASCII)
  - UTF-16 (with a BOM, for ease of identification)
  - UTF-32
- Legacy encodings (if necessary)
  - ISO 8859-1
  - Windows CP 1252
  - And many others…
- The file format should specify the encoding

More NSString Methods

- The text is in a plain-text file whose encoding is not known

```
NSStringEncoding encoding;
NSError *error;
NSString *string = [[NSString alloc] initWithContentsOfURL:url
                                        usedEncoding:&encoding
                                        error:&error];
```

NSAttributedString Reading Methods

- Reading a plain- or rich-text document

```
NSDictionary *documentAttributes;
NSError *error;
NSAttributedString *attrStr = [[NSAttributedString alloc]
    initWithURL:url options:documentOptions
    documentAttributes:&documentAttributes error:&error];
NSString *string = [attrStr string];
```

NSAttributedString Writing Methods

- Writing a plain- or rich-text document

```
NSError *error;
NSFileWrapper *fileWrapper = [attrStr
    fileWrapperFromRange:NSMakeRange(0, [attrStr length])
    documentAttributes:documentAttributes error:&error];
if (fileWrapper) {
    [fileWrapper writeToURL:url options:NSDataWritingAtomic
        originalContentsURL:nil error:&error];
}
```

String or Attributed String from Data

- Reading text data obtained from some other source

```
@interface NSString
- (id)initWithData:(NSData *)data encoding:(NSStringEncoding)enc;
- (id)initWithBytes:(const void *)bytes length:(NSUInteger)len
    encoding:(NSStringEncoding)enc;
@end

@interface NSAttributedString
- (id)initWithData:(NSData *)data options:(NSDictionary *)options
    documentAttributes:(NSDictionary **)dict error:(NSError **)err;
@end
```

String or Attributed String to Data

- Writing text data to some other destination

```
@interface NSString
- (NSData *)dataUsingEncoding:(NSStringEncoding)enc;
- (BOOL)getBytes:(void *)buffer maxLength:(NSUInteger)maxCnt
    usedLength:(NSUInteger *)usedCnt encoding:(NSStringEncoding)enc
    options:(NSStringEncodingConversionOptions)options
    range:(NSRange)range remainingRange:(NSRangePointer)leftover;
@end

@interface NSAttributedString
- (NSData *)dataFromRange:(NSRange)range
    documentAttributes:(NSDictionary *)documentAttributes
    error:(NSError **)error;
@end
```

Guessing Encodings

- Do as little guessing of encodings as possible
- Choice of encodings to try is domain-dependent
- Try UTF-8 first, then others as appropriate, finally Mac Roman
- Where appropriate, let the user intervene

Demo

Reading and writing text

Processing Text

Aki Inoue
Software Engineer

In This Section…

- Focusing on string handling tips and tricks
- How to manipulate string contents correctly and efficiently
- Inspired by questions posted to various developer forums and MLs

Technology Covered…

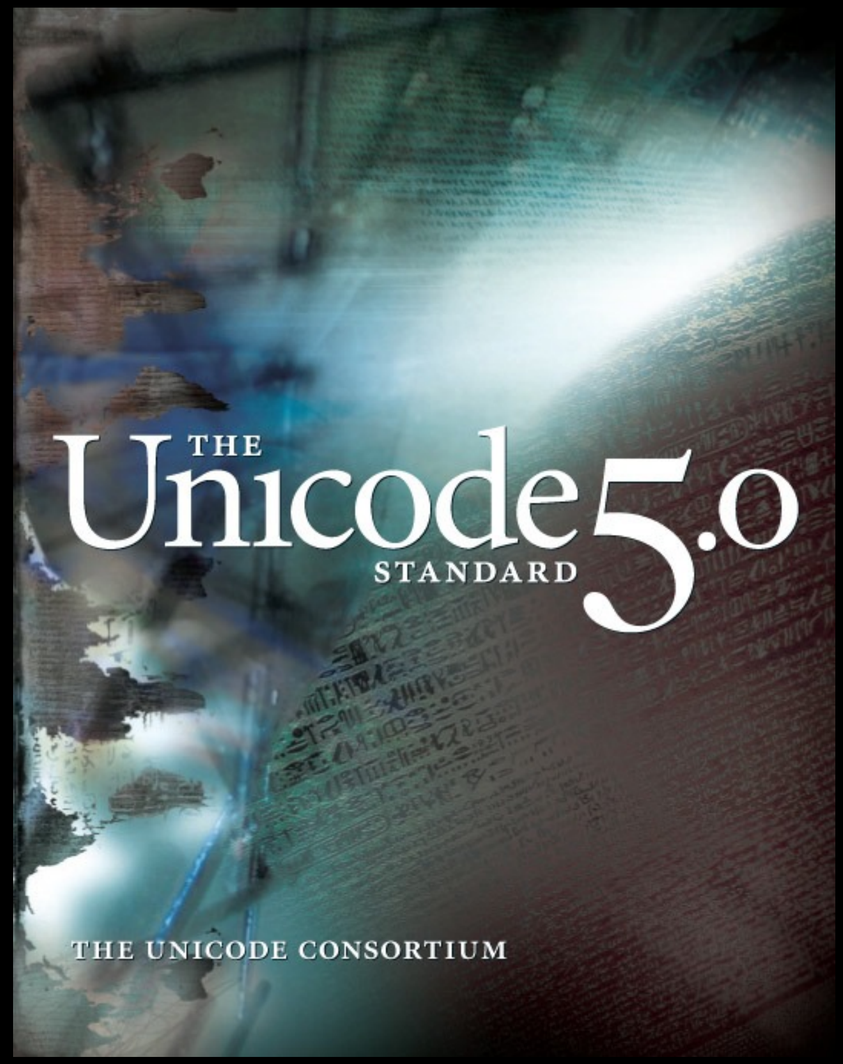
- NSString
- NSDateFormatter/NSNumberFormatter

NSString

- The string object on Mac OS X and iPhone OS
- Full-fledged string manipulation API
- Simple yet powerful
- Encapsulates Unicode

The Unicode Standard

- The universal character set
- Currently version 5.1
- 1472 page printed document
- Dozens of auxiliary reports/specifications



NSString

- The string object on Mac OS X and iPhone OS
- Full-fledged string manipulation API
- Simple yet powerful
- Encapsulates Unicode
- Insulates applications from the gigantic technical standard

What You'll Learn

- Searching and comparing
- Parsing
- Transforming
- Performance optimization

What You'll Learn

- Searching and comparing
- Parsing
- Transforming
- Performance optimization

Searching and Comparing

“My customers in Europe complain that my app doesn’t sort like Finder or Address Book.”

“How do I sort like Finder or Address Book?”

“My app doesn’t sort numerically in the localization setting.”

Searching and Comparing

- Recommend taking advantage of higher-level APIs rather than manually searching or sorting
  - Find panel
  - Spotlight
  - Core Data

Sorting

```
-compare:
-compare:options:
-compare:options:range:
-compare:options:range:locale:
```

```
-caseInsensitiveCompare:
-localizedCompare:
-localizedCaseInsensitiveCompare:
```

Sorting

-compare:options:range:locale:
-compare:options:
-compare:options:range:
-compare:options:range:locale:
-caseInsensitiveCompare:
-localizedCompare:
-localizedCaseInsensitiveCompare:

```
static NSComparisonResult MyCompareFunc(id s1, id s2, void *context) {
    NSLocale *locale = [NSLocale currentLocale];
    NSStringCompareOptions opts = NSCaseInsensitiveSearch;
    opts |= NSNumericSearch;
    opts |= NSDiacriticInsensitiveSearch;
    opts |= NSWidthInsensitiveSearch;
    opts |= NSForcedOrderingSearch;

    return [s1 compare:s2 options:opts range:NSMakeRange(0, [s1 length])
                locale:locale];
}
```

Sorting

```
-compare:options:range:locale:

static NSComparisonResult MyCompareFunc(id s1, id s2, void *context) {

    NSLocale *locale = [NSLocale currentLocale];
    NSStringCompareOptions opts = NSCaseInsensitiveSearch;
    opts |= NSNumericSearch;
    opts |= NSDiacriticInsensitiveSearch;
    opts |= NSWidthInsensitiveSearch;
    opts |= NSForcedOrderingSearch;

    return [s1 compare:s2 options:opts range:NSMakeRange(0, [s1 length])
                locale:locale];
}
```

NSStringCompareOptions

Recommended options for sorting

```
NSCaseInsensitiveSearch
NSLiteralSearch
NSBackwardsSearch
NSAnchoredSearch
NSNumericSearch
NSDiacriticInsensitiveSearch
NSWidthInsensitiveSearch
NSForcedOrderingSearch
```

fi = F + I

NSStringCompareOptions

Recommended options for sorting

```
NSCaseInsensitiveSearch
NSLiteralSearch
NSBackwardsSearch
NSAnchoredSearch
NSNumericSearch
NSDiacriticInsensitiveSearch
NSWidthInsensitiveSearch
NSForcedOrderingSearch
```

NSStringCompareOptions

Recommended options for sorting

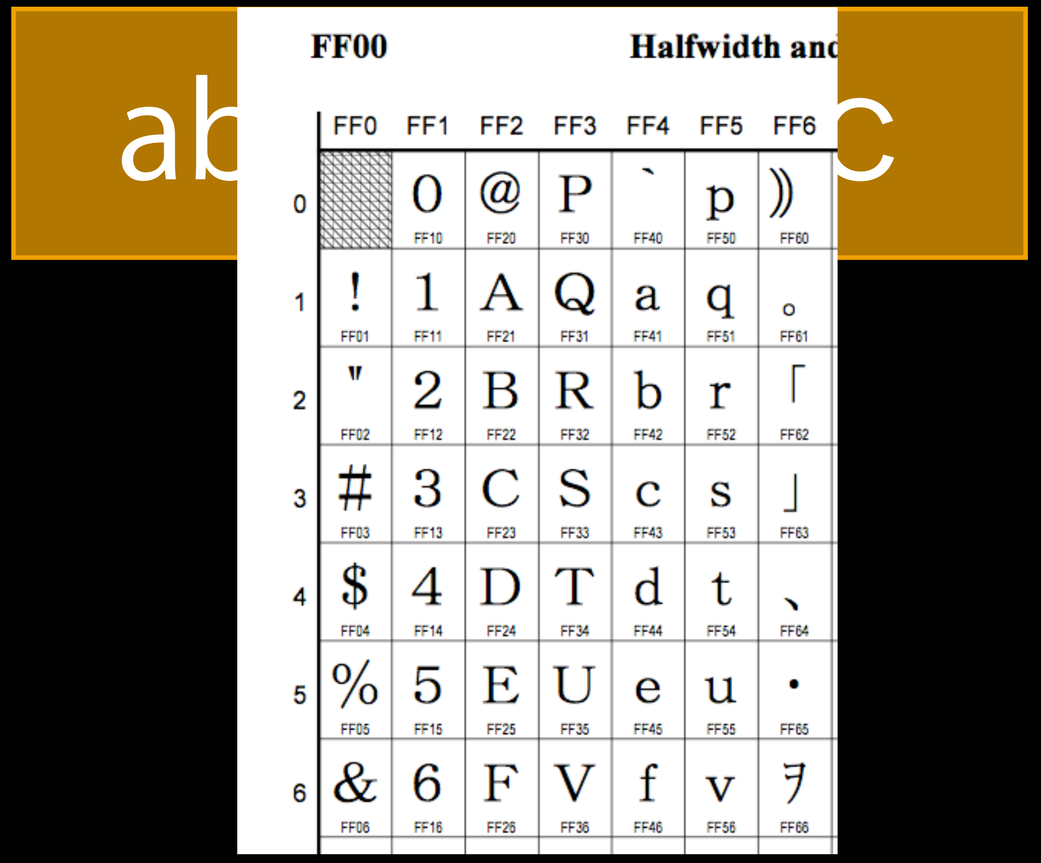
```
NSCaseInsensitiveSearch
NSLiteralSearch
NSBackwardsSearch
NSAnchoredSearch
NSNumericSearch
NSDiacriticInsensitiveSearch
NSWidthInsensitiveSearch
NSForcedOrderingSearch
```

Apres = Après

NSStringCompareOptions

Recommended options for sorting

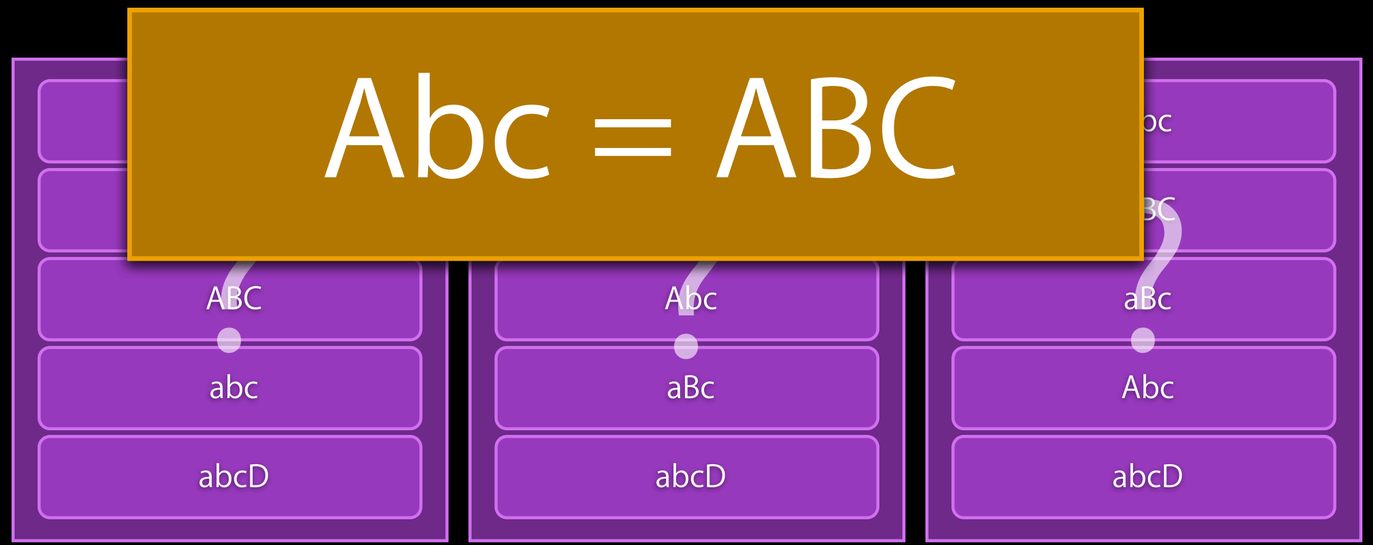
NSCaseInsensitiveSearch
NSLiteralSearch
NSBackwardsSearch
NSAnchoredSearch
NSNumericSearch
NSDiacriticInsensitiveSearch
NSWidthInsensitiveSearch
NSForcedOrderingSearch



NSStringCompareOptions

Recommended options for sorting

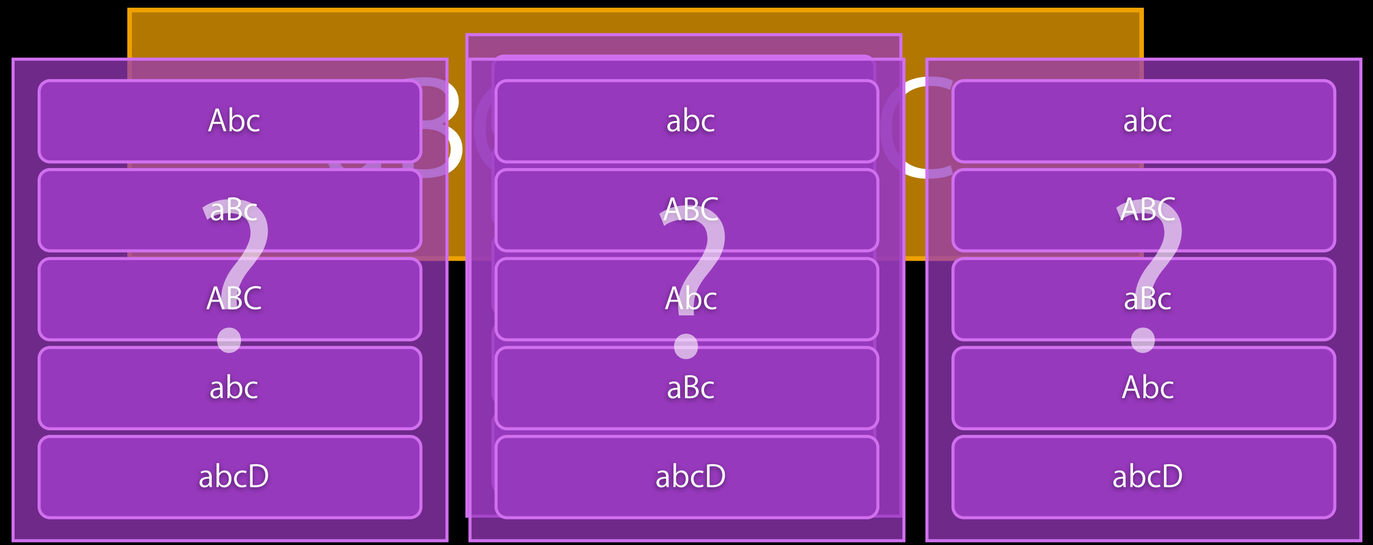
NSCaseInsensitiveSearch
NSLiteralSearch
NSBackwardsSearch
NSAnchoredSearch
NSNumericSearch
NSDiacriticInsensitiveSearch
NSWidthInsensitiveSearch
NSForcedOrderingSearch



NSStringCompareOptions

Recommended options for sorting

NSCaseInsensitiveSearch
NSLiteralSearch
NSBackwardsSearch
NSAnchoredSearch
NSNumericSearch
NSDiacriticInsensitiveSearch
NSWidthInsensitiveSearch
NSForcedOrderingSearch



Sorting

```
-compare:options:range:locale:

static NSComparisonResult MyCompareFunc(id s1, id s2, void *context) {

    NSLocale *locale = [NSLocale currentLocale];

    NSStringCompareOptions opts = NSCaseInsensitiveSearch;
    opts |= NSNumericSearch;
    opts |= NSDiacriticInsensitiveSearch;
    opts |= NSWidthInsensitiveSearch;
    opts |= NSForcedOrderingSearch;

    return [s1 compare:s2 options:opts range:NSMakeRange(0, [s1 length])
              locale:locale];
}

NSArray *sortedArray = [array sortedArrayUsingFunction:MyCompareFunc
                                               context:NULL];
```

Sorting

New in Snow Leopard

```
@interface NSString
- (NSComparisonResult)localizedStandardCompare:(NSString *)string;
@end


NSArray *sortedArray = [array sortedArrayUsingSelector:
                            @selector(localizedStandardCompare:)];
```

Filtering

“How do I filter by prefix matching?”

Filtering

```
- (BOOL)prefixMatchString:(NSString *)s1 withPattern:(NSString *)s2 {
    NSUInteger len1 = [s1 length];
    NSUInteger len2 = [s2 length];

    if (s1 < s2) return NO;

    NSLocale *locale = [NSLocale currentLocale];

    NSStringCompareOptions opts = NSCaseInsensitiveSearch;
    opts |= NSNumericSearch;
    opts |= NSDiacriticInsensitiveSearch;
    opts |= NSWidthInsensitiveSearch;

    return [s1 compare:s2 options:opts range:NSMakeRange(0, s2)
            locale:locale] == NSOrderedSame;
}
```

Filtering

```
- (BOOL)prefixMatchString:(NSString *)s1 withPattern:(NSString *)s2 {
    NSUInteger len1 = [s1 length];
    NSUInteger len2 = [s2 length];

    if (s1 < s2) return NO;

    NSLocale *locale = [NSLocale currentLocale];

    NSStringCompareOptions opts = NSCaseInsensitiveSearch;
    opts |= NSNumericSearch;
    opts |= NSDiacriticInsensitiveSearch;
    opts |= NSWidthInsensitiveSearch;

    return [s1 compare:s2 options:opts range:NSMakeRange(0, s2)
            locale:locale] == NSOrderedSame;
}
```

Not Recommended

Filtering

- When string1 and string2 are the same, it does NOT necessarily mean [string1 length] == [string2 length] because…
  - Case-insensitivity could match strings of different length
  - Accented characters could be composed or decomposed

Filtering

```
- (BOOL)prefixMatchString:(NSString *)s1 withPattern:(NSString *)s2 {
    NSUInteger len1 = [s1 length];
    NSUInteger len2 = [s2 length];

    if ((s1 == 0) || (s2 == 0)) return NO;

    NSLocale *locale = [NSLocale currentLocale];

    NSStringCompareOptions opts = NSCaseInsensitiveSearch;
    opts |= NSAnchoredSearch;
    opts |= NSNumericSearch;
    opts |= NSDiacriticInsensitiveSearch;
    opts |= NSWidthInsensitiveSearch;

    return [s1 rangeOfString:s2 options:opts range:NSMakeRange(0, s1)
            locale:locale].length > 0;
}
```

What You'll Learn

- Searching and comparing
- Parsing
- Transforming
- Performance optimization

Parsing

"It's easier for me to just extract UTF-8 C strings and process them manually."

"How do I parse external data coming from the Internet?"

"NSString is a black box to me."

Parsing

- Roughly two groups of text data commonly parsed
  - Structured
  - Unstructured (or natural language)

Structured Text Data Parsing

- Examples of common structured text data
  - HTML → DOM (WebKit)
  - XML → NSXMLParser
  - *.plist → NSPropertyList
  - *.strings files → NSBundle

Structured Text Data Parsing

- Other structured data
  - HTML/XML fragments
  - External file formats (i.e. /etc/smb.conf)
  - Structured user text input
    - Utilize NSTokenField, NSDatePicker, and/or NSFormatter subclasses

Structured Text Data Parsing

- Other structured data
  - NSScanner
  - NSString
    - -componentsSeparatedByString:
    - -componentsSeparatedByCharactersInSet:
    - -lineRangeForRange:

Structured Text Data Parsing

```
@property NSSearchField *searchField;

- (NSArray *)searchTokens {
    NSString *stringValue = [searchField stringValue];
    NSCharacterSet *separators;

    separators = [NSCharacterSet whitespaceAndNewlineCharacterSet];

    return [stringValue componentsSeparatedByCharactersInSet:separators];
}
```

Unstructured Text Processing

- Identify the right basic unit for the task
  - Grapheme clusters (a.k.a. composed character sequences)
  - Words
  - Paragraphs
- Avoid processing one character at a time

Grapheme Clusters

- The smallest processing unit for NSString
- Queried with -rangeOfComposedCharacterSequenceAtIndex:
- Composition
  - é = e + ´
- Surrogate pair
  - 丈 U+2000B = 0xD840 + 0xDC0B

Grapheme Clusters

```
NSString *string;
NSUInteger length = [string length];
NSRange range = NSMakeRange(0, 0);

while (range.location < length) {
    range = [string rangeOfComposedCharacterSequenceAtIndex:range.location];

    // do processing on the range

    range.location = NSMaxRange(range);
}
```

Words

- Appropriate processing unit for most transformation tasks
  - Letter-case mapping
  - Spell-checking
- Queried via -[NSAttributedString doubleClickAtIndex:]
- Whitespace is not necessarily the only way to break “words”
  - 正しい日本語です = 正しい + 日本語 + です
  - ภาษาไทย = ภาษา + ไทย
  - Fußballbundesliga = Fußball + bundes + liga

Words

```
NSAttributedString *attributedString;
NSUInteger length = [attributedString length];
NSRange range = NSMakeRange(0, 0);

while (range.location < length) {
    range = [attributedString doubleClickAtIndex:range.location];

    // do processing on the range

    range.location = NSMaxRange(range);
}
```

Paragraphs

- The maximum processing unit for all Unicode processing tasks
- Especially important for bidirectional languages like Arabic and Hebrew
- Each paragraph has an overall text flow direction
- Queried via -getParagraphStart:end:contentsEnd:forRange:

Paragraphs

```
NSString *string;
NSUInteger length = [string length];
NSRange range = NSMakeRange(0, 0);
NSUInteger endIndex, contentsEndIndex;

while (range.location < length) {
    [string getParagraphStart:NULL end:&endIndex
            contentsEnd:&contentsEndIndex
            forRange:NSMakeRange(range.location, 1)];

    range.length = contentsEndIndex - range.location;

    // do processing on the range

    range.location = endIndex;
}
```

Enumerating the Basic Units

New in Snow Leopard

```
@interface NSString
- (void)enumerateSubstringsInRange:(NSRange)range
        options:(NSStringEnumerationOptions)opts
        usingBlock:block;
@end
```

- Block-based API
- Enumerates line, paragraph, grapheme cluster, word, and sentence
- Line-specific convenience -enumerateLinesUsingBlock:
- New NSAttributedString methods for enumerating attribute runs

Enumerating the Basic Units

New in Snow Leopard

```
NSString *string;
NSUInteger length = [string length];
NSRange range = NSMakeRange(0, 0);
NSUInteger endIndex, contentsEndIndex;
while (range.location < length) {
    [string getParagraphStart:NULL end:&endIndex
                  contentsEnd:&contentsEndIndex
                     forRange:NSMakeRange(range.location, 1)];
    // do processing on the range
}
range.length = contentsEndIndex - range.location;

// do processing on the range

range.location = endIndex;
}
```

```
[string enumerateSubstringsInRange:NSMakeRange(0, [string length])
                           options:NSStringEnumerationByParagraphs
                        usingBlock:^(NSString *substring,
                                     NSRange substringRange,
                                     NSRange enclosingRange,
                                     BOOL *stop) {
}];
```

Demo

Space Inserter

What You'll Learn

- Searching and comparing
- Parsing
- Transforming
- Performance optimization

Transforming

“NSCalendarDate(bud, "%d", ...) with NSString?”

“NSCalendarDate(bud, "%d", ...) with NSString?”

“How do I just display the year of a date?”

Transforming

- Case mapping
  - -uppercaseString, -lowercaseString, and -capitalizedString
  - -stringByFoldingWithOptions:locale: with NSCaseInsensitiveSearch
- Normalization
  - -decomposedStringWithCanonicalMapping, etc.
- Formatting
  - NSDateFormatter and NSNumberFormatter

Number Formatting

- Traditional printf-style formatting cannot cover all patterns
  - 1,234.56
  - 1 234,56
  - ١.٢٣٤,٥٦
- NSNumberFormatter provides convenient access to common formats

NSNumberFormatter

New in Snow Leopard

```
@interface NSNumberFormatter
+ (NSString *)localizedStringFromNumber:(NSNumber *)num
                            numberStyle:(NSNumberFormatterStyle)nstyle;
@end

NSString *format = NSLocalizedString(@"%@ processed.", nil);
NSNumberFormatterStyle style = NSNumberFormatterPercentStyle;
NSString *valueString = [NSNumberFormatter localizedStringFromNumber:aValue
                                                         numberStyle:style];

message = [NSString stringWithFormat:format, valueString];
```

Date Formatting

- Recommend using one of four predefined format types
  - Short, medium, long, and full formats
- Harder to predict the final formatted string
  - Gazillions of formatting variations depending on locales
  - Users could even customize the format freely
  - Non-Gregorian calendars can be specified by the user

NSDateFormatter

New in Snow Leopard

```
@interface NSDateFormatter
+ (NSString *)localizedStringFromDate:(NSDate *)date
                            dateStyle:(NSDateFormatterStyle)dstyle
                            timeStyle:(NSDateFormatterStyle)tstyle
@end
```

```
NSDateFormatterStyle dateStyle = NSDateFormatterMediumStyle;
NSDateFormatterStyle timeStyle = NSDateFormatterNoStyle;
NSString *nowString = [NSDateFormatter localizedStringFromDate:[NSDate date]
                                                     dateStyle:dateStyle
                                                     timeStyle:timeStyle];
```

NSDateFormatter

New in Snow Leopard

```
@interface NSDateFormatter
- (BOOL)doesRelativeDateFormatting;
- (void)setDoesRelativeDateFormatting:(BOOL)b;
@end
```

NSDateFormatter

New in Snow Leopard

```
@interface NSDateFormatter
+ (NSString *)dateFormatFromTemplate:(NSString *)tmplate
    options:(NSUInteger)opts locale:(NSLocale *)locale;
@end
```

- Allows formatting dates using custom formats
- Safely formatting custom formats with “exotic” locales/calendars
- Transform custom formats to a format suitable for the locale/calendar

NSDateFormatter

New in Snow Leopard

```
@interface NSDateFormatter
+ (NSString *)dateFormatFromTemplate:(NSString *)tmplate
    options:(NSUInteger)opts locale:(NSLocale *)locale;
@end

NSString *template = @"MMM y"; // shows "Jun 2009" in US format
NSLocale *locale = [NSLocale currentLocale];
NSString *format;

// Returns "y年M月" with the Traditional Chinese locale
format = [NSDateFormatter dateFormatForTemplate:template
                                        options:0
                                         locale:locale];

NSDateFormatter *formatter = [[NSDateFormatter alloc] init];

[formatter setFormatDate:format];
```

Demo

FDFDemo

What You'll Learn

- Searching and comparing
- Parsing
- Transforming
- Performance optimization

Performance Optimization

“I really need to get dirty and need to playing with the NSString contents. What’s next?”

Performance Optimization

- The first step is always to sample with performance tools
- Majority of string manipulation perf issues come from memory allocation
  - Avoid creating many autoreleased objects; consider using mutable strings instead
  - Use CFMutableString API for transforming contents
  - Use -initWithCharactersNoCopy:length:freeWhenDone:

Things to Remember

- Unicode is NOT a mystic monster
- Use the framework APIs
- Know the appropriate processing units
  - Cluster, word, or paragraph
- Know immutable vs. mutable differences for efficient processing

Checking Text

Text Checking

New in Snow Leopard

- Automatic language identification
- Spelling and grammar checking
- Spelling correction
- Smart quotes and smart dashes
- Smart links
- Data detectors (dates, addresses)
- Text replacement
- All available automatically in NSTextView

Demo

Text checking in NSTextView

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Synchronous Checking

Check a range in a string synchronously

```
NSOrthography *orthography;
NSInteger wordCount;
NSArray *results = [[NSSpellChecker sharedSpellChecker]
    checkString:string range:rangeToCheck types:checkingTypes
    options:options inSpellDocumentWithTag:tag
    orthography:&orthography wordCount:&wordCount];
for (NSTextCheckingResult *result in results) {
    switch ([result resultType]) {
        // do something
    }
}
```

NSSpellChecker Asynchronous Checking

Check a range in a string synchronously

```
NSInteger seqNum = [[NSSpellChecker sharedSpellChecker]
    requestCheckingOfString:string range:rangeToCheck
    types:checkingTypes options:options
    inSpellDocumentWithTag:tag completionHandler:
    ^(NSInteger seqNumber, NSArray *results,
      NSOrthography *orthography, NSInteger wordCount) {
        for (NSTextCheckingResult *result in results) {
            // do something
        }
    }];
```

NSTextCheckingResult

- Represents something found in the text
  - Language identification
  - Misspelled word or grammatical error
  - URL, date, or address
  - Piece of text to be replaced (for smart quotes, dashes, correction, etc.)
- Always has type and range
- May have other information
  - Orthography
  - URL or date
  - Replacement string

Example of Text Checking

For moer information about Snow Leopard, go to www.apple.com or call 1-800-MY-APPLE.

Spelling result
+
Correction result (“more”)

Link result (http://www.apple.com)

Phone number

Overall orthography result (English language, Latin script)

NSOrthography

- Describes the linguistic content of a piece of text
- Lists scripts used, and languages for each script
- Identifies dominant script and language
- BCP-47 tags for languages and scripts
  - Languages: en, fr, de, ru, ja, ko, zh, …
  - Scripts: Latn, Cyrl, Grek, Jpan, Kore, Hans, Hant, …

Extending Spell Checking

- Learned word files in ~/Library/Spelling
  - Words separated by newlines or nulls
- System understands spelling dictionaries in MySpell/hunspell format
  - Two files, .aff and .dic
  - May be placed in any Library/Spelling directory
- Developers may write custom spelling checkers
  - Special kind of service
  - Advertise languages in their Info.plist

Demo

Text checking

More Information

Matt Drance
Application Frameworks Evangelist
mdrance@apple.com

Documentation
Strings Programming Guide
http://developer.apple.com/documentation/Cocoa/Conceptual/Strings/introStrings.html

Spell Checking
http://developer.apple.com/documentation/Cocoa/Conceptual/SpellCheck/SpellCheck.html

AppKit and Foundation Release Notes

Related Sessions

What's New in Cocoa	Marina Tuesday 9:00AM
Pasteboards, Services, and Interoperability on Mac	North Beach Thursday 2:00PM
Building Localized Mac and iPhone Applications	Presidio Friday 10:30AM

Labs

Input Method Lab	OS X Lab B Friday 2:00PM
Core Text Lab	OS X Lab B Friday 3:30PM
Mac and iPhone Internationalization Lab	iPhone Lab D Friday 3:30PM

Summary

- Handle text with NSString methods
- Work with strings and substrings
- Use Unicode encodings for external forms
- Use new text checking methods

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09